Active Learning for Dynamical Systems

Ben Sturgis
Mat. No.: 03776460

Yufei Hua
Mat. No.: 03807682

Abstract—When the model of a dynamical system is unknown, data-driven methods can be used to learn it from observed data. As collecting training data on the real physical can be costly due to experimental constraints, active learning methods can be employed to learn the model in a sample-efficient manner. A key challenge in active learning for dynamical systems is that samples cannot be arbitrarily drawn from the state space, as the system cannot be directly set to a desired state but must be guided there through an action sequence. Utilizing a Bayesian neural network to model the dynamical system, allows us to evaluate action sequences based on their informativeness through the estimated model uncertainty. We present a random sampling shooting method with model predictive control and Soft Actor Critic approach to identify informative action sequences. To ensure that the Bayesian neural network's uncertainty estimation remains independent of the input norm, we introduce a feature expansion technique. For the experiments we use two customized environments: a mass-spring-damper system and a reacher. By evaluating the one-step and multi-step prediction error for a given amount of sampled training data and additionally analyzing the state-space exploration, we demonstrate the superior sample efficiency of the active learning methods compared to random exploration. The implementation of our project is available at https://github.com/bensturgis/tum-adlr-10.

I. Introduction

Model-based control relies on an accurate mathematical representation of a dynamical system. When the model is unknown, learning-based methods can be utilized to identify the dynamics from measured data. However, a challenge for these data-driven methods is that they require a lot of training data, which, depending on the physical system, might be time-consuming and costly to collect. Active learning methods can reduce the amount of required training data by prioritizing the selection of samples which most improve the model accuracy. However, active learning for dynamical systems is challenging as we cannot arbitrarily set the system to any desired state and then test the effect of an action. Instead, we must first determine an action sequence that guides the system to that target state. But finding such a sequence requires an accurate model of the system, which is precisely what we are trying to learn.

The dynamical system is represented by a function

$$f : \mathbb{R}^{d_s} \times \mathbb{R}^{d_a} \to \mathbb{R}^{d_s}, \quad (s_t, a_t) \mapsto a_{t+1}$$

which takes the current state of the system, $s_t \in \mathbb{R}^{d_s}$, and an action, $a_t \in \mathbb{R}^{d_a}$, as inputs and outputs the next state, $s_{t+1} \in \mathbb{R}^{d_a}$, after applying action a_t in state s_t. In the following, we present and analyze methods to learn the dynamical system $f(\cdot, \cdot)$ in a sample-efficient way.

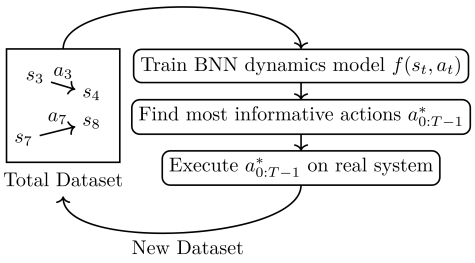


Fig. 1. Active learning for dynamical systems flowchart.

This project builds upon the methods introduced in [1], where active learning techniques for dynamical systems are extensively studied. We extend these methods by incorporating a feature expansion technique that enables uncertainty estimation in Bayesian neural networks (BNNs) independently of the input norm. This is particularly beneficial for dynamical systems, where the input cannot be simply normalized. Additionally, we evaluate the methods from [1] on a new dynamical system. The mass-spring-damper system introduces a damping and a restoring force from the spring, already studied systems in [1].

II. Active Learning for Dynamical Systems

An overview of the active learning algorithm for dynamical systems is given in Figure 1.

Starting with a small initial dataset of randomly drawn (s_t, a_t, s_{t+1}) samples, we train a BNN to learn the dynamics of the system. Using a BNN has the advantage of additionally estimating the predictive uncertainty for $f(s_t, a_t)$, which helps identifying the most informative action sequence. To measure informativeness, we use differential entropy as a metric. Assuming that the predictive distribution of $f(s_t, a_t)$ follows a Gaussian with diagonal covariance matrix $\Sigma(s_t, a_t) = \text{diag}\left(\sigma_1^2, \ldots, \sigma_{d_s}^2\right)$, its differential entropy is given by:

$$\mathcal{H}[f(s_t, a_t)] = d_s \ln(\sqrt{2\pi e}) + \frac{1}{2} \sum_{k=1}^{d_s} \ln \sigma_k^2. \tag{1}$$

The most informative action sequence $a^*_{0:T-1}$ of horizon T is found by maximizing the sum of differential entropies

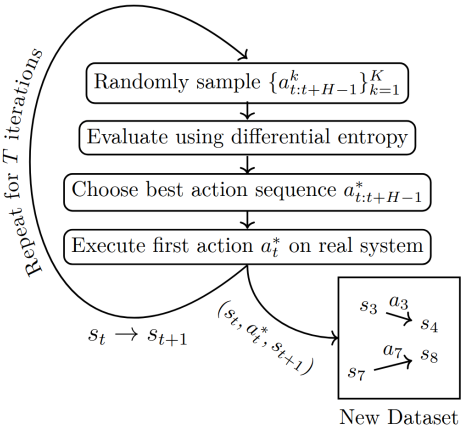


Fig. 2. Random sampling shooting with model predictive control flowchart.

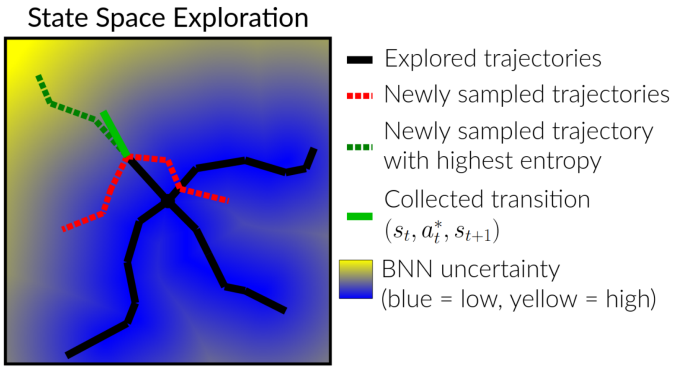


Fig. 3. Visualization of the random sampling shooting method with MPC in a two-dimensional state space.

over the expected resulting trajectory:

$$\begin{aligned} a^*_{0:T-1} = \arg\max_{a_{0:T-1}} & \sum_{t=0}^{T-1} \mathbb{E}_{s_t}\Big[\mathcal{H}\big(f(s_t, a_t)\big)\Big] \\ \text{s.t.} \quad & s_{t+1} \sim p\big(s_{t+1} \mid s_t, a_t\big), \\ & s_0 \sim p(s_0), \\ & a_{\min} \le a \le a_{\max} \end{aligned} \tag{2}$$

Once the optimal action sequence $a^*_{0:T-1}$ is found, we execute it on the real system and collect the transition samples (s_t, a_t, s_{t+1}) along the resulting trajectory. These new samples are added to our total dataset, and the process of learning the BNN and finding the most informative action sequence is iteratively repeated.

Solving the optimization problem in Equation 2 directly is intractable. Instead, we approximate the most informative action sequence using random sampling shooting combined with model predictive control (MPC) and Soft Actor Critic.

A. Random Sampling Shooting with Model Predictive Control

The flowchart in Figure 2 provides an overview of the random sampling shooting method with MPC.

In each iteration we randomly sample K action sequences $\{a^k_{t:t+H-1}\}_{k=1}^K$ with MPC planning horizon H. Starting from the current state s_t of the real system, the effect of these action sequences is then simulated using the learned model. We evaluate the informativeness of each sequence by summing the differential entropies along the expected resulting trajectory:

$$R(s_t, a_{t:t+H-1}) = \sum_{\tau=t}^{t+H-1} \mathbb{E}_{s_\tau}\left[\mathcal{H}\left[f(s_\tau, a_\tau)\right]\right]. \tag{3}$$

Since computing the expectation exactly is intractable, we approximate $R(s_t, a_{t:t+H-1})$ using Monte Carlo sampling with P trajectory rollouts:

$$R(s_t, a_{t:t+H-1}) \approx \frac{1}{P} \sum_{p=1}^{P} \sum_{\tau=t}^{t+H-1} \mathcal{H}\left[f(s^p_\tau, a_\tau)\right]. \tag{4}$$

Instead of executing the entire best action sequence $a^*_{t:t+H-1}$ on the physical system, the MPC selects and executes only the first action a^*_t which adds one transition (s_t, a_t, s_{t+1}) to our new dataset. The process is then repeated iteratively until a total of T new samples have been collected.

Figure 3 illustrates the random sampling shooting method with MPC within a two-dimensional state space.

B. Soft Actor Critic

Another approach to approximately solve the optimization problem in Equation 2 is the Soft Actor Critic, an off-policy reinforcement learning algorithm. The reward function r_t at time t is defined as the differential entropy of the BNN's predictive distribution, $\mathcal{H}[f(s_t, a_t)]$, for the input (s_t, a_t). The Soft Actor Critic is based on the maximum entropy framework. In addition to maximizing the expected reward, it also maximizes the expected differential entropy of the policy, trading-off exploration and exploitation:

$$\pi^* = \arg\max_{\pi} \sum_{t=0}^{T-1} \mathbb{E}_{s_t, a_t}\left[r_t + \alpha \mathcal{H}\left[\pi(\cdot|s_t)\right]\right]. \tag{5}$$

III. Feature Expansion for Reliable Uncertainty Estimation

Upon implementing the uncertainty map visualization, we identified a problem in the uncertainty estimation. In BNNs, the predictive variance is inherently proportional to the input norm, making uncertainty comparisons across inputs with different norms invalid.

Proposition 1. Consider a neural network with one hidden layer of h neurons and ReLU activation, where the parameters are initialized from $\mathcal{N}(0, \sigma^2)$. For a given

deterministic input $\mathbf{x}$, Bayesian inference has the following output variance:

$$\mathrm{Var}(y_k) = \frac{h\sigma^4}{2}\|\mathbf{x}\|_2^2$$

Thus, the variance of the output scales proportionally to $\|\mathbf{x}\|^2$.

Proof. For a detailed proof, see Appendix A. □

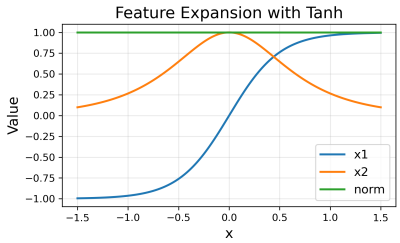


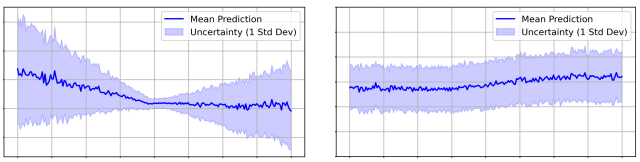
Fig. 4. Illustration of the tanh feature expansion.

We solved this problem using a nonlinear feature expansion. To ensure uniform initial uncertainty across the input space, the input norm should remain constant. The key idea is to introduce additional features that compensate for the original input norm without compromising model capacity. Assuming that original input $\mathbf{x}$ is loosely constrained by $[-\mathbf{B}, \mathbf{B}]$ construct non-linear feature transformation

$$\mathbf{x}_{\text{exp}} = \begin{bmatrix} \tanh\left(\frac{2\mathbf{x}}{\mathbf{B}}\right) \\ \sqrt{1 - \tanh^2\left(\frac{2\mathbf{x}}{\mathbf{B}}\right)} \end{bmatrix}$$

ensuring that $\|\mathbf{x}_{\text{exp}}\|^2 = 1$. Refer to Figure 4, which illustrates the transformed features and there norm with $B = 1$.

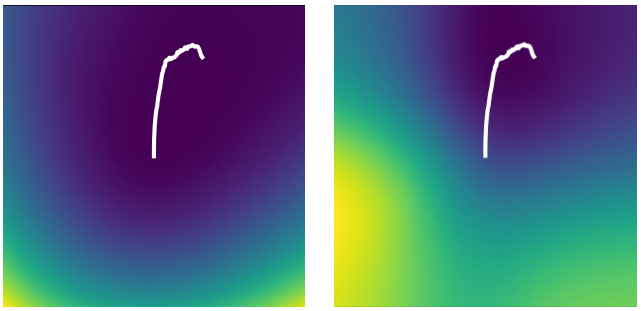
To test the feature expansion on a BNN with scalar input and output we plot the network's predictive variance, which represents its uncertainty, upon random initialization of the network. In Figure 5(a), the original feature is used, and the uncertainty is proportional to the input value. In Figure 5(b), expanded features are applied, and the uncertainty distributes uniformly. To



(a) Original features. (b) Expanded features.

Fig. 5. Predictive variance upon initialization.

test the feature expansion trick on a mass-spring-damper system, we plot the uncertainty across the two-dimensional state space after training on a template trajectory. In Figure 6(a), the network with the original feature shows a radially increasing pattern, indicating that the uncertainty is dominated by the input norm. In Figure 6(b), the network with expanded features can predict more reasonable uncertainty, with the low uncertainty region appearing only around the trajectory.



(a) Original features. (b) Expanded features.

Fig. 6. Uncertainty map after one training iteration. (blue: low, yellow: high)

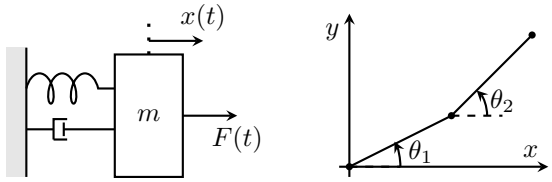
IV. Experiments and Results

A. Environments

We compared active learning algorithms on two environments, which are the mass-spring-damper system (Figure 7(a)) and reacher environment (Figure 7(b)).

The mass-spring-damper system consists of a mass m connected to a fixed wall by a spring and a damper. The state consists of the position and velocity, represented as $\begin{bmatrix} x & \dot{x} \end{bmatrix}^\top$, while the action is the applied force, given by $\begin{bmatrix} F \end{bmatrix}^\top$.

The reacher system is a robotic arm with two joints [2]. The state consists of the cosine- and sine-representation of the joint angles θ_1 and θ_2, as well as their angular velocities, represented as $\begin{bmatrix} \cos\theta_1 & \cos\theta_2 & \sin\theta_1 & \sin\theta_2 & \dot{\theta}_1 & \dot{\theta}_2 \end{bmatrix}^\top$, while the action consists of the torques applied at each joint, given by $\begin{bmatrix} a_1 & a_2 \end{bmatrix}^\top$.



(a) Mass-spring-damper system. (b) Reacher.

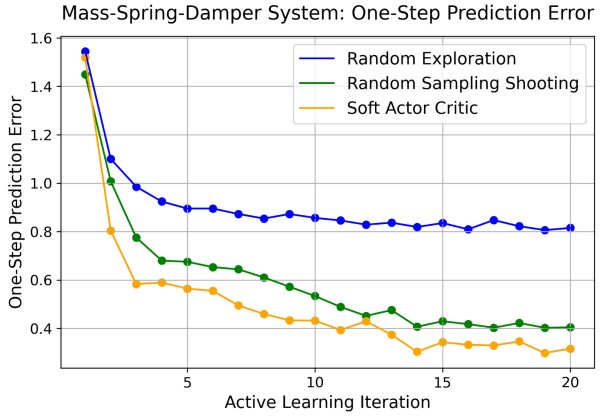
Fig. 7. Simulated dynamical systems.

B. Prediction Error Curves

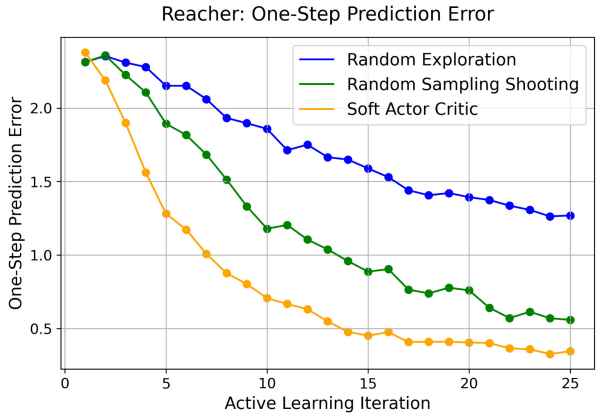
To evaluate the effectiveness of active learning algorithms, we evaluated two metrics: One-Step Prediction Error and Multi-Step Prediction Error.

One-step prediction error measures how accurately the model predicts a single-step transition in the environment.

The evaluation process involves randomly sampling a set of state-action pairs, executing their one-step transitions in real environment, and using the trained model to predict the next states. The prediction error is then computed as the difference between the model's predictions and the actual observed transitions. The results, presented in Figures 8(a) and 8(b), show that active learning algorithms significantly reduce the prediction error by collecting more informative training samples.



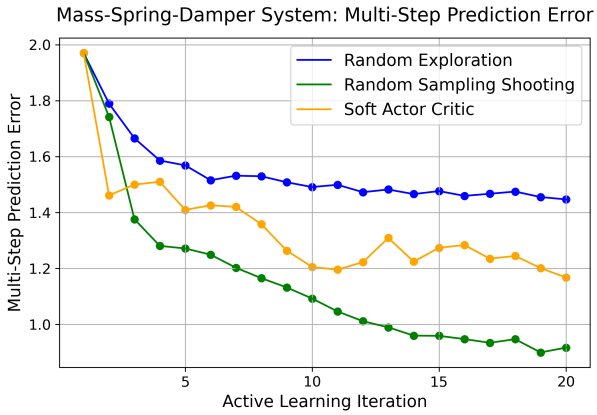
(a) Mass-Spring-Damper System.



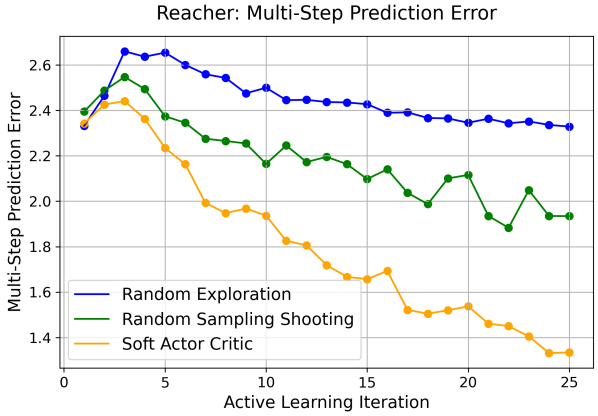
(b) Reacher.

Fig. 8. One-step prediction error curve.

Multi-step prediction error evaluates the model's ability to predict multiple time steps trajectories. The process involves collecting short trajectories from the real system, selecting random states along these trajectories as starting points, and predicting multiple future states ahead using the trained model with the same action sequences as input. The the error of the last predicted state is calculated. In our experiments, the Mass-Spring-Damper system predicted 10 steps ahead, while the Reacher environment predicted 7 steps ahead. The results, shown in Figures 9(a) and 9(b), illustrate that by employing active learning methods less data is required to reduce the multi-step prediction error.



(a) Mass-Spring-Damper System.



(b) Reacher.

Fig. 9. Multi-step prediction error curve.

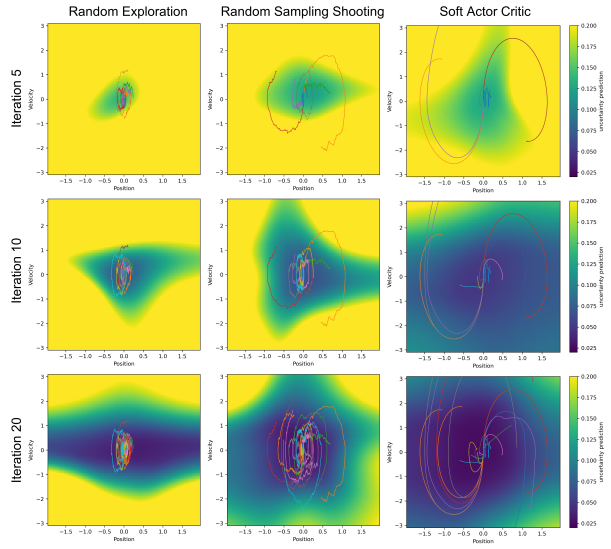
The results indicate that active learning algorithms significantly improve sample efficiency by collecting more informative samples. This allows the neural network to learn system dynamics more effectively with fewer data samples. As shown in Figures 8(a) to 9(b), AL algorithms demonstrate faster error reduction compared to baseline exploration strategies such as random exploration.

C. Uncertainty Maps

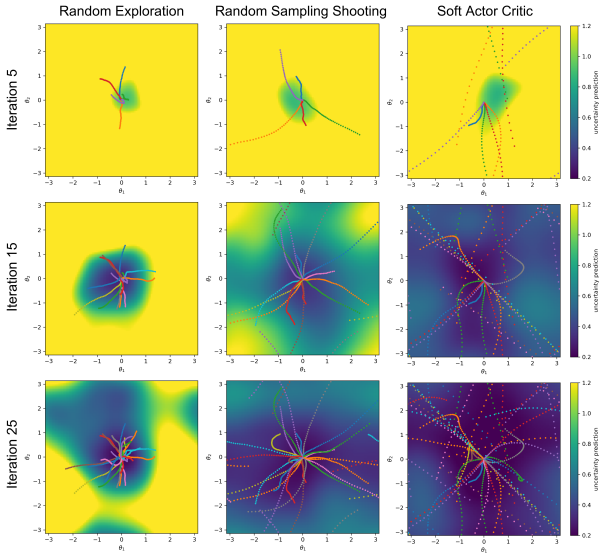
To better understand how the uncertainty-based active learning algorithms work, we visualize the uncertainty distribution across a two-dimensional state space, as shown in Figure 10. The uncertainty maps illustrate how confident the BNN think it is in different regions of the state space. These maps are color-coded, where yellow indicates high uncertainty, and blue represents high confidence. As the model gathers more informative data, the confident regions (blue areas) are expected to expand over iterations.

Across both systems, the uncertainty maps show a clear trend of increasing confidence regions as the model trains over multiple iterations.

Random Exploration leaves large areas of high uncertainty after the whole learning process, as trajectories stay



(a) Mass-Spring-Damper System.



(b) Reacher.

Fig. 10. Uncertainty maps of both environments after certain iterations of active learning, comparing random exploration, random sampling shooting with MPC, and Soft Actor Critic.

close to the origin with limited exploration.

Random sampling shooting improves state space coverage, allowing the confident region to expand. Because of the inherent randomness, trajectories collected by this method are still noisy, and sometimes fail to reach the most informative region after several sub-optimal action choice.

Among all methods, Soft Actor Critic achieves the most thorough exploration, reducing uncertainty more effectively while producing smoother and more trajectories.

D. Bayesian neural network: Laplace Approximation vs. Monte-Carlo Dropout

Initially, we used Monte-Carlo Dropout due to its ease of implementation. However, we observed that MC Dropout was highly sensitive to the dropout probability, often leading to less accurate predictions. This prompted us to explore alternative realization for BNNs.

We then implemented Laplace Approximation [3]. In this method, posterior distributions of parameters are estimated by additionally computing Hessian matrices, which means the training and inference can be conducted exactly the same as normal fully connected neural networks. In system dynamic prediction tasks, it also allows solely deterministic inference.

As as result, Laplace Approximation is particularly advantageous for our dynamic learning task involving small networks. By computing the Hessian matrices of the parameters in the last layer, the model can still provide reliable uncertainty predictions. Moreover, it is more computationally efficient during testing, as only deterministic state prediction is required.

V. Conclusion

This project explored active learning methods to learn dynamical systems in a sample-efficient way. We introduced a random sampling shooting method with MPC and a Soft Actor Critic approach to identify informative action sequences based on the uncertainty estimates of the BNN modeling the system. Our experiments demonstrate that both methods significantly reduce the required training data while simultaneously improving model accuracy for the investigated dynamical systems, with the Soft Actor Critic achieving even more comprehensive state-space exploration than the random sampling shooting with MPC. In future work, we aim to explore novel evaluation metrics to better evaluate the sample efficiency of active learning algorithms and develop a systematic approach for generating test sets.

References

[1] Shengbing Tang, Kenji Fujimoto, and Ichiro Maruta. Actively learning dynamical systems using bayesian neural networks. Applied Intelligence, 53: 29338–29362, 2023.

[2] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John Balis, Gianluca Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Tai, Hannah Tan, and Omar Younis. Gymnasium: A Standard Interface for Reinforcement Learning Environments. arXiv.2407.17032, 2024.

[3] Hanlin Yu, Marcelo Hartmann, Bernardo Williams, Mark Girolami, and Arto Klami. "Riemannian Laplace Approximation with the Fisher Metric." arXiv preprint arXiv:2311.02766, 2023. AISTATS 2024, with additional fixes and improvements. DOI: 10.48550/arXiv.2311.02766.

Appendix

Proof of Proposition 1

Proof. Consider a neural network with one hidden layer containing h neurons. The activation function used is ReLU. Define the parameter matrices W and W_0 as:

$$W = \begin{bmatrix}\vec{w}_1 & \vec{w}_2 & \dots & \vec{w}_h\end{bmatrix}^T, \qquad W \in \mathbb{R}^{h\times m}$$
$$W_0 = \begin{bmatrix}\vec{w}_{o1} & \vec{w}_{o2}\end{bmatrix}^T, \qquad W_0 \in \mathbb{R}^{n\times h}$$

where the elements of W and W_0 are drawn independently from $\mathcal{N}(0,\sigma^2)$.

Given a deterministic input $\vec{x}$, the output of the network is given by:

$$\vec{y} = W_0\,\mathrm{ReLU}(W\vec{x})$$

We will derive the output distribution in three steps.

Step 1: Single Neuron in the Hidden Layer

Define the pre-activation value for a single neuron in the hidden layer:

$$z_{1,\ell} = \sum_i w_{\ell,i}x_i, \quad \ell = 1,2,\dots,h$$

Since $w_{\ell,i} \sim \mathcal{N}(0,\sigma^2)$, we compute the expectation $E[z_{1,\ell}] = 0$ and variance:

$$\begin{aligned}E[z_{1,\ell}^2] &= E\left[\sum_i w_{\ell,i}^2 x_i^2\right] = \sum_i x_i^2 E[w_{\ell,i}^2]\\ &= \sum_i x_i^2\sigma^2 = \|\mathbf{x}\|^2\sigma^2\end{aligned}$$

Thus, $z_{1,\ell} \sim \mathcal{N}(0, \|\vec{x}\|^2\sigma^2)$.

Step 2: Applying ReLU Activation

After applying ReLU to $z_{1,\ell}$:

$$z_{2,\ell} = \mathrm{ReLU}(z_{1,\ell}), \quad \ell = 1,2,\dots,h$$

the probability density function (PDF) of $z_{2,\ell}$ is:

$$P(z_{2,\ell}|\vec{x}) = \begin{cases}\frac{1}{2}, & z_{2,\ell} = 0\\ \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{z_{2,\ell}^2}{2\sigma^2}}, & z_{2,\ell} > 0\end{cases}$$

The expectation and second moment of $z_{2,\ell}$ are:

$$\begin{aligned}E[z_{2,\ell}] &= \int_0^\infty zP(z|\vec{x})dz = \|\vec{x}\|\sigma\frac{1}{\sqrt{2\pi}}\\ E[z_{2,\ell}^2] &= \int_0^\infty z^2P(z|\vec{x})dz = \frac{\|\vec{x}\|^2\sigma^2}{2}\left(1-\frac{1}{\pi}\right).\end{aligned}$$

Step 3: Computing Output Variance

Define the final output:

$$y_k = \sum_i w_{k,i}z_{2,i}, \quad k = 1,2,\dots,n$$

Since $E[z_{2,i}]$ is nonzero, we compute:

$$E[y_k] = E\left[\sum_i w_{k,i}z_{2,i}\right] = \sum_i E[w_{k,i}]E[z_{2,i}] = 0.$$

Now, compute the variance:

$$\begin{aligned}E[y_k^2] &= E\left[\left(\sum_i w_{k,i}z_{2,i}\right)^2\right]\\ &= \sum_i E\left[(w_{k,i}z_{2,i})^2\right]\\ &= \sum_i \mathrm{Var}(w_{k,i}z_{2,i})\end{aligned}$$

Using the formula for variance of independent products:

$\mathrm{Var}(XY) = E[X]^2\mathrm{Var}(Y)+E[Y]^2\mathrm{Var}(X)+\mathrm{Var}(X)\mathrm{Var}(Y)$

and substituting values:

$$\begin{aligned}E[w_{k,i}^2]E[z_{2,i}^2] + \sigma^2E[z_{2,i}^2] &= \sum_i\left(\frac{\|\vec{x}\|^2\sigma^4}{2\pi} + \sigma^2\frac{\|\vec{x}\|^2\sigma^2}{2}(1-\frac{1}{\pi})\right)\\ &= \sum_i\frac{\|\vec{x}\|^2\sigma^4}{2}\left(1+\frac{1}{\pi}-1\right)\\ &= \frac{h\sigma^4}{2}\|\vec{x}\|^2.\end{aligned}$$

Thus, the variance of the output scales proportionally to $\|\vec{x}\|^2$:

$$\mathrm{Var}(y_k) \propto \|\vec{x}\|^2.$$

□